**Practical Assignment – 6**

1. **Write a python program to implement k-means algorithm to build prediction model (Use Credit Card Dataset CC GENERAL.csv Download from kaggle.com)**

import pandas as pd

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import MinMaxScaler

from sklearn.cluster import KMeans

# Load data

df = pd.read\_csv("CC GENERAL.csv")

# Drop ID column

df = df.drop(columns=["CUST\_ID"])

# Fill missing values with median

imputer = SimpleImputer(strategy="median")

df\_imputed = pd.DataFrame(imputer.fit\_transform(df), columns=df.columns)

# Scale features to 0-1 range

scaler = MinMaxScaler()

df\_scaled = pd.DataFrame(scaler.fit\_transform(df\_imputed), columns=df.columns)

# Run KMeans with 5 clusters (you can change this number)

kmeans = KMeans(n\_clusters=5, random\_state=42)

df\_scaled["Cluster"] = kmeans.fit\_predict(df\_scaled)

# Add cluster labels back to original data (optional)

df["Cluster"] = df\_scaled["Cluster"]

# Show count of points in each cluster

print(df["Cluster"].value\_counts())

1. **Write a python program to implement hierarchical Agglomerative clustering algorithm. (Download Customer.csv dataset from github.com).**

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from sklearn.cluster import AgglomerativeClustering

import matplotlib.pyplot as plt

from scipy.cluster.hierarchy import dendrogram, linkage

# Load dataset

df = pd.read\_csv("Customers.csv")

# Preview dataset (optional)

print(df.head())

# Drop non-numeric or ID columns if any (adjust column names accordingly)

# For example, if there's 'CustomerID' or 'Name' column, drop them:

df\_numeric = df.select\_dtypes(include=[np.number]) # keep only numeric columns

# Handle missing values if any (simple median imputation)

df\_numeric = df\_numeric.fillna(df\_numeric.median())

# Scale features 0-1

scaler = MinMaxScaler()

data\_scaled = scaler.fit\_transform(df\_numeric)

# Plot dendrogram to help decide number of clusters

linked = linkage(data\_scaled, method='ward')

plt.figure(figsize=(10, 7))

dendrogram(linked,

orientation='top',

distance\_sort='descending',

show\_leaf\_counts=False)

plt.title('Dendrogram')

plt.xlabel('Samples')

plt.ylabel('Distance')

plt.show()

# Choose number of clusters, e.g. 3

n\_clusters = 3

agg\_clust = AgglomerativeClustering(n\_clusters=n\_clusters, affinity='euclidean', linkage='ward')

labels = agg\_clust.fit\_predict(data\_scaled)

# Add cluster labels back to original dataframe

df['Cluster'] = labels

# Print cluster counts

print(df['Cluster'].value\_counts())

# Optional: save to CSV

df.to\_csv("Customer\_with\_clusters.csv", index=False)

print("Clustering complete. Results saved to Customer\_with\_clusters.csv")

1. **Write a python program to implement k-means algorithms on a synthetic dataset.**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

# \*\*Step 1: Generate Synthetic Dataset\*\*

n\_samples = 300

n\_features = 2

n\_clusters = 4

random\_state = 42

X, y = make\_blobs(n\_samples=n\_samples, n\_features=n\_features, centers=n\_clusters, random\_state=random\_state)

# \*\*Step 2: Visualize the Dataset\*\*

plt.scatter(X[:, 0], X[:, 1], c='gray', s=30)

plt.title("Synthetic Dataset")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.show()

# \*\*Step 3: Apply K-Means Algorithm\*\*

kmeans = KMeans(n\_clusters=n\_clusters, random\_state=random\_state)

kmeans.fit(X)

# \*\*Step 4: Visualize Clusters\*\*

plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels\_, cmap='viridis', s=30)

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], c='red', marker='X', s=200, label='Centroids')

plt.title("K-Means Clustering")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.legend()

plt.show()

1. **Write a python program to implement hierarchical clustering algorithm. (Download Wholesale customers data dataset from github.com).**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

# Step 1: Load the dataset

url = "Wholesale customers data.csv"

df = pd.read\_csv(url)

# Step 2: Drop non-numeric or categorical columns if needed

data = df.drop(columns=['Channel', 'Region'])

# Step 3: Standardize the data

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(data)

# Step 4: Perform Hierarchical Clustering

linked = linkage(scaled\_data, method='ward') # You can try 'single', 'complete', etc.

# Step 5: Plot the dendrogram

plt.figure(figsize=(10, 5))

dendrogram(linked, truncate\_mode='lastp', p=10)

plt.title('Hierarchical Clustering Dendrogram (last 10 merges)')

plt.xlabel('Sample index or cluster size')

plt.ylabel('Distance')

plt.show()

# Step 6: Create clusters

num\_clusters = 4

cluster\_labels = fcluster(linked, num\_clusters, criterion='maxclust')

# Step 7: Add cluster labels to DataFrame

df['Cluster'] = cluster\_labels

print(df[['Cluster']].value\_counts())

# Step 8: Plot clusters using two original features (e.g., 'Grocery' vs 'Milk')

plt.figure(figsize=(8, 6))

plt.scatter(data['Grocery'], data['Milk'], c=cluster\_labels, cmap='tab10', s=50)

plt.title('Clusters based on Grocery and Milk')

plt.xlabel('Grocery')

plt.ylabel('Milk')

plt.grid(True)

plt.show()